From Fault-tolerance to Attack Tolerance

AFOSR Grant F9550-06-1-0019

Final Report

1 December 2005 – 30 November 2010

Fred B. Schneider
Computer Science Department
Cornell University
Ithaca, New York
(607) 255-9221 (phone)
fbs@cs.cornell.edu

1 Objectives

Means to build fault-tolerant services have been at hand for some time. Defense against attacks remains a difficult problem, though. And this problem becomes ever more urgent with the increasing use of networked computing systems in our society's critical infrastructures and in future-generation military systems (such as GIG and JBI). The objective of this AFOSR-funded effort is to bridge the gap from fault-tolerance to attack-tolerance by exploring two threads.

- The use of mechanically-generated diversity for creating independent server replicas and a "moving target" defense.
- Language-based techniques to build a new theoretical basis for authorization and for quantifying information flow and information corruption.

2 Summary of Completed Research

2.1 Moving target defenses: Theory and practice

Semantic Framework for Diversity. A set of replicas is diverse to the extent that all implement the same functionality but differ in their imple-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
02 APR 2011	**FINAL REPORT**	**00-01-2005 to 30-11-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
From Fault-tolerance to Attack Tolerance	**FA9550-06-1-0019**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Fred Schneider	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Cornell University,Ithaca,NY,14853	**; AFRL-OSR-VA-TR-11-043**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/RSL, 875 Randolph Street, Suite 325, Room 3112, Arlington, VA, 22203	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **AFRL-OSR-VA-TR-11-043**

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Means to build fault-tolerant services have been at hand for some time. Defense against attacks remains a difficult problem, though.The problem becomes ever more urgent with the increasing use of networked computing systems in our society's critical infrastructures and in future-generation military systems (such as GIG and JBI). The objective of this AFOSR-funded effort was to bridge the gap from fault-tolerance to attack-tolerance by exploring two threads. The first thread was to explore the use of mechanically-generated diversity for creating independent server replicas and a ''moving target'' defense. This led to a implementing a prototype system that embodied our proactive obfuscation scheme and to a theory that establishes mechanically-generated diversity is almost as powerful a defense as typechecking. The second thread was to explore language-based techniques and build a new theoretical basis for authorization and for quantifying information flow and information corruption. Here, Nexus Authorization Logic (NAL) was developed and deployed it as part of a new operating system.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**9**	

mentation details. Diverse replicas are less prone to having vulnerabilities in common, because attacks typically depend on memory layout and/or instruction-sequence specifics.

Recent work advocates using mechanical means, such as program rewriting, to create such diversity. A correspondence between the specific transformations being employed and the attacks they defend against is often provided. But little has been said about the overall effectiveness of diversity per se in defending against attacks. With this broader goal in mind, we developed a precise characterization of attacks, applicable to viewing diversity as a defense. In addition, we showed how mechanically-generated diversity compares to a well-understood defense: strong typing.

The reduction we derived—defenses created by mechanically-generated diversity are forms of probabilistic dynamic type-checking—was surprising. Unfortunately, this result ignores probabilities, which do matter for practical application. The work is thus best seen as only a first step in characterizing the effectiveness of program obfuscation and other forms of mechanically-generated diversity.

Proactive Obfuscation Prototype. Proactive obfuscation is a moving-target defense. It involves running diverse replicas and periodically selecting a replica for replacement by a new one, where that new one differs internally from any prior replica used in the system. Proactive obfuscation creates independence and also preserves it over time, even when attackers can analyze individual replicas.

A firewall controls the passage of packets from some outer network to an enclave it is protecting. Because it resides at the border to a potentially hostile network, attack-tolerance is crucial for a firewall. And because it is the sole means by which packets enter or exit an enclave, availability is also important. A firewall was thus an ideal service to implement using an approach, like proactive obfuscation, intended for building trustworthy services.

We built that prototype and conducted a rather extensive analysis of its performance. This analysis involved deploying different implementations of the various underlying mechanisms, so that we could identify bottlenecks. We also implemented a distributed storage service that uses Byzantine Quorum Systems (rather than state machine replication) and employs proactive obfuscation to create the artificial diversity needed for independence among those servers.

Convincing Alternative to Digital Signatures. Systems that employ proactive obfuscation and, indeed, most protocols for fault-tolerance and attack-tolerance, often rely on public-key signatures for message-source authentication, defense against corruption of in-transit messages, and also as a proof to recipients of correct protocol execution. In our proactive obfuscation prototypes, as in many distributed services, the overhead of generating and checking signatures must be kept low. With public-key signatures it won't be, which prompted us to develop lower-cost alternatives for this setting.

A *k-convincing tag* can be forwarded at least $k-1$ times and still be guaranteed to convince receivers of its authenticity. We developed an $O(knlogn)$ probabilistic protocol for the case when n hosts might receive such a tag; in some contexts, our protocol is faster than using public-key signatures. We also developed an $O(n^2\ log^2 n)$ probabilistic protocol for computing k-convincing tags for all k simultaneously. And we ran experiments to compare the performance of these protocols to fast implementations of public-key signatures and to closely related MAC constructions.

2.2 Language-based techniques

Hyperproperties. Important classes of *security policies* cannot be expressed using what have been termed *properties*, sets of execution traces for which membership of a trace depends on the trace alone and not on which other traces are in the property. For example, *noninterference* is a confidentiality policy that stipulates commands executed on behalf of users holding high clearances have no effect on system behavior observed by users with only low clearances. Noninterference is not a property, because whether some given trace is allowed depends on whether another trace (obtained by deleting command executions by high users) is allowed. As a second example, stipulating a bound on average response time over all executions is an availability policy that cannot be specified as a property, because the acceptability of delays in any given execution depends on the magnitude of delays in all other executions.

These expressiveness limitations are overcome by using a new abstraction we developed—*hyperproperties*, sets of properties (i.e., sets of sets of traces). There are two interesting classes of hyperproperties: safety and liveness. And we have been able prove the following.

- Hyperproperties can describe properties and, moreover, can describe security policies, such as noninterference and average response time,

that properties cannot. Indeed, we have not been able to find requirements on system behavior that cannot be specified as a hyperproperty. Deterministic, nondeterministic, and probabilistic system models all can be handled using hyperproperties.

- Every hyperproperty is the intersection of a safety hyperproperty and a liveness hyperproperty. Safety hyperproperties and liveness hyperproperties thus form a fundamental basis from which all hyperproperties can be constructed.

- The topological characterization of properties can be generalized to characterize hyperproperties, and the result is equivalent to the *lower Vietoris* topology.

We have not been able to obtain complete verification methods for safety hyperproperties or for liveness hyperproperties, but we have been able to generalize prior work on using invariance arguments to verify information-flow policies. Our generalization is applicable to a class of hyperproperties we introduce called *k-safety*.

We have also been able to relate the hyperproperties framework to *stepwise refinement*. Whereas safety properties are preserved by such refinement steps, hypersafety properties are not. We also explored the extent to which the hyperproperties framework applies to arbitrary system representations, such as relations, labeled transition systems, and probabilistic state machines.

Quantification of Integrity. Hyperproperties are qualitative. In fact, the usual characterization of security in terms of confidentiality, integrity, and availability is qualitative. Engineering realities often require quantitative characterizations. Methods have long existed for specifying and verifying quantitative bounds on the flow of confidential information. Yet methods for quantification of *corruption*—that is, damage to integrity—have received little attention to date.

Under the auspices of this funding, we developed a framework and a method to calculate bounds on integrity corruption. To quantify corruption, a formal definition of "integrity" was required. We took two distinct notions of information modification as points of departure: taint analysis and program correctness. These, in turn, led to two distinct measures of corruption that we named *contamination* and *suppression*.

Contamination is defined to be the flow of information from untrusted inputs to outputs that are supposed to be trusted. Trusted outputs are

not supposed to be influenced by untrusted information, so contamination corrupts integrity. Flow between untrusted and trusted objects was first studied by Biba, who identified a duality between models of integrity and confidentiality. The confidentiality dual to contamination is *leakage*, which is information flow from secret inputs to public outputs. The Biba duality thus suggests that any method for measuring leakage of information could serve as the basis for measuring contamination.

Our other measure for corruption, suppression, is derived from program correctness. For a given input, a correct implementation should produce an output o permitted by a specification. Any knowledgeable user of these implementations could recover o from the implementation's output. With programs and channels, suppression occurs when information is lost. Information theory can be used to quantify suppression, including how to bound the attacker's influence on suppression.

We might suspect that contamination generalizes suppression, or vice versa, but we have proved this is not the case. Moreover, we have been able to use our approach to derive quantitative measures for various anonymization algorithms that have been proposed to support database privacy: k-anonymity, l-diversity, and differential privacy. This work is based on a new theorem we derived; it asserts that suppression plus leakage is necessarily a constant (related to the information content of the object being anonymized).

Credentials-based Authorization. Authorization is fundamental to implementing any trustworthy system. To be *trustworthy*, a system must behave as expected but not exhibit any other behaviors. And authorization, which governs what requests a system will accept, is the way requests for unacceptable system behaviors are blocked.

Nexus Authorization Logic (NAL) was developed to provide a principled basis for specifying and reasoning about credentials and authorization policies. It extended prior access control logics based on "says" and "speaks-for" operators, enabling within a single framework request authorization to depend on (i) the source or pedigree of the requester, (ii) the outcome of performing an analysis on the requester, or (iii) the use of trusted software to encapsulate or modify the requester. Prototype document-viewer applications that enforce integrity and confidentiality of document contents—all implemented on the Nexus operating system—have been built to illustrate the convenience and expressive power of this approach to authorization.

3 Impacts on the Community

It can be difficult to trace the impact that ideas have, and the research reported herein is mostly concerned with new ideas. However, there are two direct consequences that can likely be traced to our research. They concern framing of the problem space (and ultimately new federal funding).

- The CNCI "moving target defense" focus area apparently had its origins with the proactive obfuscation work discussed above.
- A MURI and a new CNCI initiative on *science of security*, including a DDRE-funded Jasons study (JSR-10-102) in Summer 2010, all are direct responses to the PI's advocacy (in Congressional testimony and in discussions with DoD). The language-based technique discussed above has provided examples of the pay-off these investments could provide.

There are other less-direct transitions that arise through the PI's involvement in various advisory capacities.

- Schneider is Chief Scientist of the NSF TRUST Science and Technology Center, which includes U.C. Berkeley, Carnegie-Mellon University, Cornell University, Stanford University, and Vanderbilt University.
- Schneider is a member of the following industrial advisory boards: Fortify Software Technical Advisory Board; Microsoft's Trustworthy Computing Academic Advisory Board (co-chair).
- Schneider served on the following other advisory committees: NIST Information Security and Privacy Advisory Board; Computing Research Association Board of Directors; Defense Science Board; Computing Community Consortium Council.

4 Publications Supported

1. Kevin Hamlen, Greg Morrisett, and Fred B. Schneider. Computability classes for enforcement mechanisms. *TOPLAS 28*, 1 (January 2006), 175–205.
2. Kevin Hamlen, Greg Morrisett, and Fred B. Schneider. Certified Inlined Reference Monitoring on .NET. *Proceedings ACM SIGPLAN Workshop on Programming Languages and Analysis for Security* (Ottawa, Canada, June 2006), 7–16.

3. Fred B. Schneider. Here Be Dragons. Editorial. *IEEE Security and Privacy*, Volume 4, Number 3 (May/June 2006), 3.

4. Riccardo Pucella and Fred B. Schneider. Independence from obfuscation: A semantic framework for diversity. *Proceedings 19th IEEE Computer Security Foundations Workshop* (Venice, Italy, July 2006), IEEE Press, 2006, 230–241.

5. Fred B. Schneider. Trusted computing in context. Editorial. *IEEE Security and Privacy*, Volume 5, Number 2 (March/April 2007), 4–5.

6. Fred B. Schneider, Dexter Kozen, Greg Morrisett, and Andrew C. Myers. Language-Based Security for Malicious Mobile Code. *Department of Defense Sponsored Information Security Research: New Methods for Protecting Against Cyber Threats*, Wiley Publishing Company, Indianapolis, Indiana, 2007, 477–494.

7. Fred B. Schneider. Credentials-Based Authorization: Evaluation and Implementation. Abstract of Plenary Lecture. *Proceedings 34th International Colloquium*, ICALP 2007 (Wroclaw, Poland, July 2007), Lecture Notes in Computer Science, Volume 4596, Springer-Verlag, Heidelberg, 2007, 12–14.

8. Fred B. Schneider. Mapping the Security Landscape: A Role for Language Techniques. Abstract of Invited Lecture. *Proceedings 18th International Conference, CONCUR 2007* (Lisbon, Portugal, September 2007), Luis Caires and Vasco T. Vasconcelos (eds.). Lecture Notes in Computer Science, Volume 4703, Springer-Verlag, Heidelberg, 2007, 1.

9. Yee Jiun Song, Robert van Renesse, Fred B. Schneider, and Danny Dolev. The building blocks of consensus. *Proceedings 9th International Conference on Distributed Computing and Networking* ICDCN 08, (Kolkata, India, Jan. 2008), Lecture Notes in Computer Science, Volume 4904 (S. Rao et al, eds.), Springer-Verlag, Heidelberg, 2008, 54–72.

10. Michael Clarkson and Fred B. Schneider. Hyperproperties. *Proceedings 21st IEEE Computer Security Foundations Symposium* (Pittsburgh, PA, June 2008), 51–65.

11. Dan Williams, Patrick Reynolds, Kevin Walsh, Emin Gun Sirer, and Fred B. Schneider. Device driver safety through a reference validation

mechanism. *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation* OSDI '08, (San Diego, CA, December 2008), 241–254.

12. Fred B. Schneider and Ken Birman. The monoculture risk put into context. *IEEE Security and Privacy* Volume 7, Number 1 (January/ February 2009), 14–17.

13. Stefan Savage and Fred B. Schneider. Security is not a commodity: The road forward for cybersecurity research. Computing Research Initiatives for the 21st Century, Computing Community Consortium. February 2009. http://www.cra.org/ccc/initiatives .

14. Aaron Burstein and Fred B. Schneider. Trustworthiness as a limitation on network neutrality. *Federal Communications Law Journal*, Vol. 61 (2009), 591–623.

15. Fred B. Schneider. Accountability for Perfection. Editorial. *IEEE Security and Privacy* Volume 7, Number 2 (March/April 2009), 3–4.

16. Ken Birman and Fred B. Schneider. The PC Overload Problem. *Communications of the ACM 52*, 05 (May 2009), 34–37.

17. Fred B. Schneider. Testimony for Hearing on Cyber Security R & D. United States House of Representatives Committee on Science and Technology, Research and Science Education Subcommittee, hearing June 10, 2009 on Cyber Security R & D.

18. Fred B. Schneider. Testimony. United States House of Representatives Committee on Science and Technology, Technology and Innovation Subcommittee, hearing October 22, 2009 on Cybersecurity Activities at NIST's Information Technology Laboratory.

19. Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying Information Flow with Beliefs. Invited paper. *Journal of Computer Security* 17(5), 655–701, 2009.

20. Fred B. Schneider. Testimony. United States House of Representatives Armed Services Committee, Terrorism, Unconventional Threats, and Capabilities Subcommittee, hearing February 25, 2010 on Private Sector Perspectives on Department of Defense Information Technology and Cybersecurity Activities.

21. Tom Roeder and Fred B. Schneider. Proactive Obfuscation. *ACM Transactions on Computer Systems* 28, 2 (July 2010), 1–54.

22. Michael Clarkson and Fred B. Schneider. Quantification of Integrity. *Proceedings 23rd IEEE Computer Security Foundations Symposium* CSF 2010, (Edinburgh, UK, July 2010), 28–43.

23. Fred B. Schneider. Fumbling the Future, Again. Editorial. *IEEE Security and Privacy* Volume 8, Number 4 (July/August 2010), 3.

24. Riccardo Pucella and Fred B. Schneider. Independence from Obfuscation: A Semantic Framework for Diversity. *Journal of Computer Security* 18, 5 (August 2010), 701–749.

25. Michael R. Clarkson and Fred B. Schneider. Hyperproperties. Invited paper. *Journal of Computer Security* 18, 6 (September 2010), 1157–1210.